Flow ambiguity: A path towards classically driven blind quantum computation

Atul Mantri,[1,2] Tommaso F. Demarie,[1,2] Nicolas C. Menicucci,[3,4] and Joseph F. Fitzsimons[1,2]

[1]*Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372*
[2]*Centre for Quantum Technologies, National University of Singapore, Block S15, 3 Science Drive 2, Singapore 117543*
[3]*School of Science, RMIT University, Melbourne, Victoria 3001, Australia*
[4]*School of Physics, The University of Sydney, Sydney, New South Wales 2006, Australia*

It is very likely that when a universal quantum computer will finally become available, it will be hosted by large institutions and accessed remotely by clients. For example, companies like D-Wave [1] and IBM [2], as well as academic institutions including the University of Bristol [3], have begun making their quantum devices available for remote access. This situation will inevitably lead to questions as to the integrity and privacy of the client's computation. In the past, quantum protocols have been proposed to address similar problems. Protocols which provide security of client's quantum computation, as well as input and output, are known as blind quantum computing protocols [4–7]. Similarly, protocols which capture the idea of verification of quantum computing, i.e., the ability to detect, with very high probability, any attempt by a malicious server to deviate from the computation are known as verifiable quantum computing protocols [8–10]. A common feature among known protocols for these tasks is that either the client require a small quantum device on their side or there must exist at least two non-communicating quantum servers [11, 12]. In other words, there is a requirement that two or more parties involved in the protocol possess quantum processors. Ideally, we would like to have a secure delegated quantum computing protocol between a completely classical client and a quantum server. In this work, we take the first steps towards this problem. We construct a blind quantum computing protocol which maintains security of the client's computation even against the quantum server. In the next section, we briefly describe our main ideas and results, but the full details can be found in [13].

Main Ideas and Results

Our aim is to explore the possibility of blind quantum computation with a purely classical client. We demonstrate this fact by constructing a protocol for a task we call as classically driven blind quantum computing (CDBQC) and analyse its security in the stand-alone setting. Our protocol uses measurement-based quantum computing (MBQC) [14] as the underlying principle. We show that the protocol allows a client to hide non-trivial information about their computation from the powerful quantum server by making use of a novel technique that we call flow ambiguity. In particular, we analyse the case of a single instance of the protocol and show that the amount of information obtained by the server is bounded below what is necessary to unambiguously distinguish the computation.

In the MBQC framework we denote by Δ the computation of the client such that $\Delta = \{\mathcal{G}, \boldsymbol{\alpha}, \boldsymbol{f}\}$. Here $\mathcal{G}$ denotes the graph state, $\boldsymbol{\alpha}$ is the set of measurement angles on the graph state, and $\boldsymbol{f}$ represents the information flow [15] which captures how angles are to be adapted based on results of previous measurements. Formally, generalised information flow or g-flow [16] is defined as follows: For an open graph $\mathcal{G}(I, O)$, there exists a *g-flow* $(g, \succ)$ if one can define a function $g : O^c \to P(I^c)$ and a partial order $\succ$ on $\mathcal{V}$ such that $\forall i \in O^c$, all of the following conditions hold:

(G1) if $j \in g(i)$ and $j \neq i$, then $j \succ i$;
(G2) if $j \not\succ i$ and $i \neq j$, then $j \notin \mathrm{Odd}(g(i))$; and
(G3) $i \notin g(i)$ and $i \in \mathrm{Odd}(g(i))$.

Intuitively, g-flow is used to assign a set of local corrections to a subset of unmeasured qubits to ensure deterministic computation, despite the random nature of the measurement outcomes obtained during the computation. It is important to note that for a fixed graph there exist multiple choices of the input and output vertex sets that result in deterministic measurement patterns consistent with the same fixed total ordering of vertices. Specifically, we show that the transcript of any run of the protocol is consistent with multiple non-equivalent computations. This is due to the fact that the information about the g-flow for the underlying resource state is hidden from the server. This particular ambiguity in the flow enables the classical user to hide the essential aspects of the computation.

The CDBQC protocol is interactive and proceeds as follows. Firstly, the client sends the dimension of the graph to the server to prepare the graph state $|G\rangle$. At each step i: Client chooses a bit r_i uniformly random. Using r_i and the previous measurement outcome $b_{<i'}$, client updates the angle α_i to construct α'_i in the following way:

$$\alpha' = (-1)^{s^x}\alpha + (r \oplus s^z)\pi$$

where s^x and s^z denote the corrections dictated by flow based on previous measurement results. The server performs a projective measurement of i-th vertex in the XY-plane of the Bloch sphere, denoted $M_i^{\alpha_i'} = \{|\pm_{\alpha_i'}\rangle\langle\pm_{\alpha_i'}|\}$, where $\{|\pm_{\alpha'}\rangle\} = \frac{1}{\sqrt{2}}(|0\rangle \pm e^{i\alpha'}|1\rangle)$ and sends the measurement outcome b_i' to the client. The client records $b_i = b_i' \oplus r_i$ in $\boldsymbol{b}$ and then updates the set $(\boldsymbol{s}^x, \boldsymbol{s}^z)$. If the i-th vertex is output qubit then the bit b_i is registered in the set $\boldsymbol{p}_B^C$. The client and the server repeat this procedure for all the vertex of the graph in the given total order. The client implements the final round of corrections on the string $\boldsymbol{p}^C$ (equivalent to $\boldsymbol{p}_B^C$ in the case of honest server) to obtain the output string $\boldsymbol{p}$. At the end of the protocol, the server possesses information about the angles $\boldsymbol{\alpha'}$ and the measurement outcome $\boldsymbol{b'}$ and whereas the client's secret consists of the actual measurement angle $\boldsymbol{\alpha}$ and the flow bits $\boldsymbol{f}$. Hereafter we will denote the variables with the upper-case letters and particular instances of such variable with the lower-case letters. For example, $\boldsymbol{A}$ will be used to denote the angle variable and $\boldsymbol{F}$ is used to represent the flow variable. For simplicity, let's take the case when $\boldsymbol{A}$ and $\boldsymbol{F}$ are uniformly variables. We quantify the amount of information that on average remains hidden from the server about the client's computation at the end of the protocol. This is given by the conditional entropy $H(\boldsymbol{A}, \boldsymbol{F}|\boldsymbol{B'}, \boldsymbol{A'})$.

$$H(\boldsymbol{A}, \boldsymbol{F}|\boldsymbol{B'}, \boldsymbol{A'}) = H(\boldsymbol{A}, \boldsymbol{F}) - I(\boldsymbol{B'}, \boldsymbol{A'}; \boldsymbol{A}, \boldsymbol{F}). \tag{1}$$

where $H(\boldsymbol{A}, \boldsymbol{F}) = H(\boldsymbol{A}) + H(\boldsymbol{F}) := \log_2 N_{\boldsymbol{A}} + \log_2 N_{\boldsymbol{F}}$. Here $N_{\boldsymbol{A}}$ and $N_{\boldsymbol{F}}$ denote the number of possible choices for the angle and flow variable respectively. Using tools from information theory we explicitly calculate that, in a single run of CDBQC protocol, the mutual information between the client's secret input ($\boldsymbol{\alpha}$, $\boldsymbol{f}$) and the information received by the server ($\boldsymbol{\alpha'}$, $\boldsymbol{b'}$) is bounded by

$$I(\boldsymbol{B'}, \boldsymbol{A'}; \boldsymbol{A}, \boldsymbol{F}) \leq H(\boldsymbol{A'})$$

This in turn gives a lower bound on the conditional entropy

$$H(\boldsymbol{A}, \boldsymbol{F}|\boldsymbol{B'}, \boldsymbol{A'}) \geq \log_2 N_{\boldsymbol{F}} \tag{2}$$

We derive a non-trivial lower bound on the conditional entropy by calculating the value of $N_{\boldsymbol{F}}$. Note that flow is a property of the underlying graph and therefore depends on the chosen graph $\mathcal{G}$. We will consider the case of cluster states as they are known to be universal for quantum computation with (X, Y)-plane measurements [17]. To calculate $N_{\boldsymbol{F}}$ for the cluster state, we put a lower bound on the number of different input and output choices (open graphs) $\#\mathcal{G}(I, O)_{n,m}$ satisfying flow conditions for a cluster state and a certain fixed total order. Mathematically, this corresponds to calculating the flows that satisfy the conditions (G1)-(G3) as mentioned above. To simplify the counting argument we put an additional constraint:

(G4) If $k \in \mathcal{N}(i) \cup \mathcal{N}(j)$, and if $k \in g(i)$, then $k \notin g(j)$.

This is not required strictly by the definition of g-flow, but it simplifies the flow counting problem and so we obtain a lower bound on the number of flows rather than the exact number. For a generic cluster state $\mathcal{G}_{(n,m)}$ with the fixed total ordering of measurements, the number of different open graphs $G(I, O)$ satisfying the conditions (G1)- (G4) is given by:

$$\#\mathcal{G}(I, O)_{n,m} = F_{2\min(n,m)+1}^{|n-m|} \prod_{\mu=2}^{\min(n,m)} F_{2\mu}^2. \tag{3}$$

where F_i is the ith Fibonacci number. Further simplifying the above equation gives us $\#\mathcal{G}(I, O)_{n,m} = 2^{2N\log_2\phi + O(N^\epsilon)}$ for $\epsilon < 1$, $N = nm$ and assuming $m = \text{poly}(n)$. This shows that there exists at least an exponential number (in the dimension of the graph) of information flows corresponding to a cluster state for a given total order of measurements. To demonstrate this we take a simple example of 2×2 cluster state $G(I, O)_{(2\times 2)}$ in Figure 1. The figure shows 9 possible open graphs compatible with the flow conditions (G1)-(G4). In general different flows correspond to different computations.

Using the following relation $N_{\boldsymbol{F}} \geq \#\mathcal{G}(I, O)_{n,m}$ with the the above result, we get $\log_2 N_{\boldsymbol{F}} \geq \log_2 \#\mathcal{G}(I, O)_{n,m} \approx 1.388N$. Therefore, the conditional entropy is given by

$$H(\boldsymbol{A}, \boldsymbol{F}|\boldsymbol{B'}, \boldsymbol{A'}) \geq 1.388N. \tag{4}$$

This shows that it is indeed possible for a client to hide their chosen computation, by using the ambiguity in the flow of information, from a quantum server. Importantly, we show that it is not possible for the quantum server to

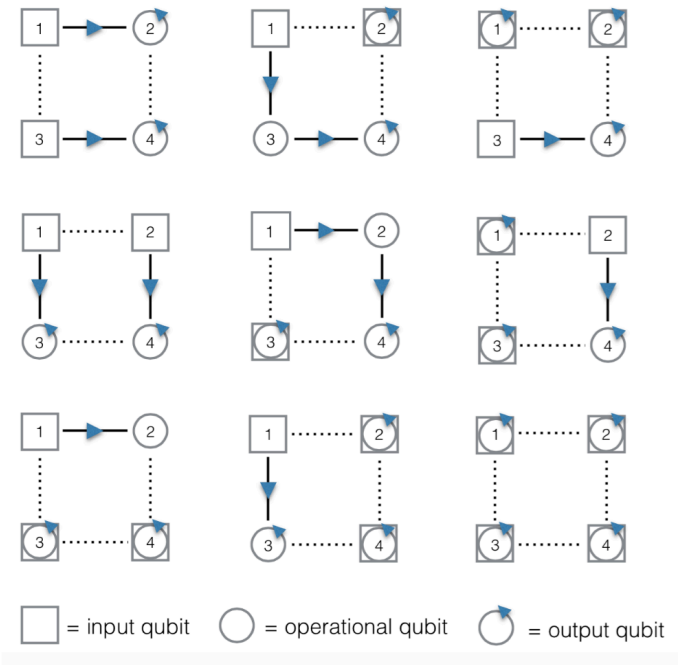

FIG. 1: All the possible $\mathcal{G}(I,O)_{2,2}$ combinations that satisfy g-flow conditions for the 2×2 cluster state are shown. The arrows indicate the direction of the quantum information flow. Note that overlapping input and output sets are allowed. All patterns implement unitary embeddings on the input state.

guess the client's computation perfectly, since a large number of other computations are still compatible with the information server receives.

For more details, we refer to the arxiv version of this work [13] and references therein.

[1] D-Wave. https://www.dwavesys.com/
[2] IBM. The quantum experience. URL http://www.research.ibm.com/quantum/
[3] Quantum in the Cloud. http://www.bristol.ac.uk/physics/research/quantum/engagement/qcloud/
[4] Childs, A. M. (2005). Secure assisted quantum computation. Quantum Information & Computation, 5(6), 456-466.
[5] Arrighi, P., & Salvail, L. (2006). Blind quantum computation. International Journal of Quantum Information, 4(05), 883-898.
[6] Broadbent, A., Fitzsimons, J., & Kashefi, E. (2009, October). Universal blind quantum computation. In Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on (pp. 517-526). IEEE.
[7] Morimae, T., & Fujii, K. (2013). Blind quantum computation protocol in which Alice only makes measurements. Physical Review A, 87(5), 050301.
[8] Fitzsimons, J. F., & Kashefi, E. (2012). Unconditionally verifiable blind computation. arXiv preprint arXiv:1203.5217.
[9] Broadbent, A. (2015). How to verify a quantum computation. arXiv preprint arXiv:1509.09180.
[10] Hayashi, M., & Morimae, T. (2015). Verifiable measurement-only blind quantum computing with stabilizer testing. Physical review letters, 115(22), 220502.
[11] Reichardt, B. W., Unger, F., & Vazirani, U. (2013). Classical command of quantum systems. Nature, 496(7446), 456-460.
[12] Fitzsimons, J. F. (2016). Private quantum computation: An introduction to blind quantum computing and related protocols. arXiv preprint arXiv:1611.10107.
[13] Mantri, A., Demarie, T. F., Menicucci, N. C., & Fitzsimons, J. F. (2016). Flow ambiguity: A path towards classically driven blind quantum computation. arXiv preprint arXiv:1608.04633.
[14] Raussendorf, R., & Briegel, H. J. (2001). A one-way quantum computer. Physical Review Letters, 86(22), 5188.
[15] Danos, V., & Kashefi, E. (2006). Determinism in the one-way model. Physical Review A, 74(5), 052310.
[16] Browne, D. E., Kashefi, E., Mhalla, M., & Perdrix, S. (2007). Generalized flow and determinism in measurement-based quantum computation. New Journal of Physics, 9(8), 250.
[17] Mantri, A., Demarie, T. F., & Fitzsimons, J. F. (2017). Universality of quantum computation with cluster states and (X, Y)-plane measurements. Scientific Reports, 7.